Ocean Dynamics (2015) 65:1743–1760
DOI 10.1007/s10236-015-0893-z

The all-source Green's function (ASGF) and its applications to storm surge modeling, part I: from the governing equations to the ASGF convolution

Zhigang Xu [1]

Received: 30 October 2014 / Accepted: 29 September 2015 / Published online: 16 November 2015

Abstract In this study, a new method of storm surge modeling is proposed. This method is orders of magnitude faster than the traditional method within the linear dynamics framework. The tremendous enhancement of the computational efficiency results from the use of a pre-calculated all-source Green's function (ASGF), which connects a point of interest (POI) to the rest of the world ocean. Once the ASGF has been pre-calculated, it can be repeatedly used to quickly produce a time series of a storm surge at the POI. Using the ASGF, storm surge modeling can be simplified as its convolution with an atmospheric forcing field. If the ASGF is prepared with the global ocean as the model domain, the output of the convolution is free of the effects of artificial open-water boundary conditions. Being the first part of this study, this paper presents mathematical derivations from the linearized and depth-averaged shallow-water equations to the ASGF convolution, establishes various auxiliary concepts that will be useful throughout the study, and interprets the meaning of the ASGF from different perspectives. This paves the way for the ASGF convolution to be further developed as a data-assimilative regression model in part II. Five Appendixes provide additional details about the algorithm and the MATLAB functions.

This article is part of the Topical Collection on the *6th International Workshop on Modeling the Ocean (IWMO) in Halifax, Nova Scotia, Canada 23–27 June 2014*

Responsible Editor: Kevin Lamb

✉ Zhigang Xu
Zhigang.Xu@dfo-mpo.gc.ca

[1] Fisheries and Oceans Canada, Maurice Lamontagne Institute, Mont-Joli, Quebec, Canada

Keywords Shallow-water equations · Storm surges · The all-source Green's functions (ASGFs) · Convolution

1 Introduction

This paper presents a new method for modeling storm surges called the all-source Green's function (ASGF) method. In contrast to the traditional Green's function, where only one or a few grid points can be the source points, the ASGF allows all of the model grid points to be its source points. The ASGF was first proposed by Xu (2007) to instantaneously predict the arrival times and wave amplitudes of a tsunami at a point of interest (POI) from an arbitrary tsunami source region (see also Xu 2011 and Xu and Song 2013). The ASGF can also be used to efficiently model storm surges and tides; however, this paper focuses on its application to storm surge modeling.

The next section will demonstrate that the ASGF can be numerically derived from a storm surge model. All of the numerical features of the storm surge model are passed to the ASGF. The solution that is obtained using the ASGF method at a POI will be identical to or practically the same as the solution that is obtained for the same point by running the surge model traditionally (cf. Eqs. (26) and (27) and the last paragraph of Section 3.2). However, the ASGF method can compute the solution orders of magnitude faster because it eliminates the computations at all the grid points where the solutions are not of interest; the method focuses its computations at only one or a few points where the solutions are desired. The traditional method must map out the solutions at all of the grid points regardless of whether they are needed. As a result of this great improvement in computational efficiency, very-long-term simulations become feasible. For example, it is desirable to hydrodynamically convert various existing century-long climate model solutions to storm surge time

series for coastal risk assessments due to climate change. Such long-term simulations might be infeasible with the traditional method but can be achieved in a few minutes using the ASGF method. Xu et al. (2015) recently demonstrated how the ASGF method was used to efficiently produce 140 years of hourly time series of storm surges driven by past and future climate forcing spanning from 1961 to 2100.

In addition to being fast, the ASGF method accounts for the influences of global forcing fields and the global ocean geometry because the ASGF can be affordably prepared with the global oceans as the model domain. This bypasses the need to address artificial open-water boundary issues. Theoretically, something that occurs at any location in the world ocean will eventually affect the solution at the POI significantly or insignificantly (cf. Section 4). Regardless of the significance of the global influences, it is better to include the entire world ocean as the domain to calculate the ASGF because that can make artificial open-water boundaries unnecessary.

Historically, the Green's functions for the linearized and depth-averaged shallow-water equation (SWE) were used to model storm surges by Welander (1961), Uusitalo (1960), and Schwab (1978) and to model tides by Munk and Cartwright (1966). A common feature shared by the Green's functions used in these studies is that the source points of the Green's functions were limited to one or a few grid points. The source points are the grid points where the external forcing field is distributed. In reality, the atmospheric forcing or the tide-generating forcing field is distributed throughout the model domain, not only at one or a few points. This source point limitation problem arises from the ways how these Green's functions were obtained. The Green's functions were obtained empirically by analyzing the historical response data at a POI and the forcing data at the same point (Munk and Cartwright 1966) and at a few additional points elsewhere (Welander 1961). The number of source points was therefore limited by how many such data analyses could be easily conducted. In Uusitalo (1960) and Schwab (1978), the Green's functions were dynamically calculated with a numerical model but in a traditional way, as briefly described in the following: one places an impulse at a grid point and then runs the model for a period of time to obtain the solution as a Green's function; one then relocates the impulse to another grid point and reruns the model to obtain another Green's function; to obtain a complete set of Green's functions, one would have to repeat this procedure for all of the grid points, which is not feasible when the number of grid points is large. Consequently, Schwab (1978) had to limit his Green's function computations to a few wind stations available in his model domain (Lake Eric). Uusitalo (1960) limited his Green's function calculations to only two model runs; each run was for a constant and domain-wise uniform wind stress applied in one of two orthogonal principal directions. By taking the time derivatives of the solutions from the two model runs, he obtained two Green's functions, with which he could then model storm surges in his model domain (the Baltic Sea) driven by any wind stress that could vary arbitrarily in time but must be uniform in space. Thus, by assuming the spatial uniformity in two principal directions, he equivalently reduced the source points to two special "points". Obviously, the challenge from the real world is that the wind is not spatially uniform, especially over a large domain. In general, the traditional method of calculating the Green's functions always leads to a source limitation problem in one way or another. The ASGF algorithm is a new method of calculating the Green's functions. As will be seen in Section 3, this algorithm completely eliminates the source limitation problem; one model run can include all of the grid points as the source points. In the context of tsunami problems, Xu and Song (2013) also discussed the differences between the ASGF and the block-source Green's functions (BSGFs).

As with any other type of Green's function, the ASGF can only be applied to linear dynamic systems. However, linear dynamics provide a first-order approximation (e.g., Pedlosky 1979), especially for storm surges (e.g., Welander 1961; Heaps 1969), tides (e.g., Laplace 1776; Lamb 1932; Munk and Cartwright 1966; Randall 2007), and tsunamis in deep water (e.g., Shuto 1991). For locations where nonlinear effects may be strong, one may establish a local nonlinear model with its open-water boundaries set at locations where the nonlinearity is expected to be weak. In such cases, the ASGF method can be used to provide a nonlinear model with open-water boundary conditions, in terms of the barotropic components, by supplying the time series of the sea surface elevations and water mass transports along the open-water boundaries. This topic will be explored in a future study.

This study consists of two parts. This paper is part I and is mainly devoted to the development of the ASGF convolution from the depth-averaged linear SWE. Part II will be presented in the next paper (Xu 2015), which will further develop the ASGF convolution to a regression model for data assimilation. This paper is organized as follows: Section 2 establishes a canonical matrix equation for the depth-averaged linear SWE. Using the matrix equation, Section 3 defines the ASGF and presents the corresponding algorithm. This section also introduces two auxiliary concepts, the memory time scale and the sampling rate, which the subsequent development and analyses, especially those in part II, will depend on. Section 4 interprets the ASGF in terms of the domain of dependence of the wave solutions at a point, in terms of the response functions to the impulses at all of the grid points, and in term of the system control language. Section 5 summarizes the paper. In addition, five appendixes provide complementary details about the algorithm and the MATLAB functions. Testing of the algorithm with analytical solutions and with real a storm surge event and an analysis of the computational efficiency of the new method will be presented in part II.

2 Linear and depth-averaged SWE in a matrix form

In Pedlosky (1979), one can find a systematic order analysis showing that linear terms in the SWE are the leading balancing terms for large-scale geophysical fluid dynamics. Welander (1961) performed the same type of order analysis pertinent to storm surge problems and concluded that as long as the characteristic amplitude of the surge is small compared with the water depth and as long as the characteristic horizontal scale is large compared with the water depth, the nonlinear terms can be neglected from the SWE. Because the two conditions are generally met in the ocean, the depth-averaged linear SWE have commonly been adopted in many studies on storm surges (e.g., Proudman 1954; Svansson 1959; Uusitalo 1960; Welander 1961; Heaps 1969; Schwab 1978). This linearization will also be well supported by the results of a real storm surge case in this study, which will be presented in part II (Xu 2015). The purpose of this study is to provide a new method to quickly model storm surges within the linear dynamical framework. Therefore, the following depth-averaged linear SWE in matrix form is chosen as the starting point for this study:

$$\frac{\partial}{\partial t}\begin{bmatrix}\eta\\ U\\ V\end{bmatrix} = -\begin{bmatrix} 0 & \dfrac{\partial}{\cos\varphi\ \partial x} & \dfrac{\partial \cos\varphi}{\cos\varphi\ \partial y}\\ gh\dfrac{\partial}{\cos\varphi\ \partial x} & \dfrac{\kappa}{h} & -f\\ gh\dfrac{\partial}{\partial y} & f & \dfrac{\kappa}{h}\end{bmatrix}\begin{bmatrix}\eta\\ U\\ V\end{bmatrix} + \begin{bmatrix} 0 & 0 & 0\\ gh\dfrac{\partial}{\cos\varphi\ \partial x} & 1 & 0\\ gh\dfrac{\partial}{\partial y} & 0 & 1\end{bmatrix}\begin{bmatrix}\eta_a\\ \tau_x\\ \tau_y\end{bmatrix} \tag{1}$$

where t is time; λ, φ, and R denote the longitude, latitude and the Earth's mean radius R (taken as 6371 km), respectively; x and y are the arc lengths along the equator and a meridian circle ($x=R\lambda$ and $y=R\varphi$), respectively; $\frac{\partial}{\partial x}=\frac{\partial}{R\partial\lambda}$; $\frac{\partial}{\partial y}=\frac{\partial}{R\partial\varphi}$; η, U, and V are the sea surface elevation and the mass fluxes[1] in the longitudinal and latitudinal directions; f and g are the Coriolis parameter and gravity acceleration, respectively; and h and κ are the water depth and bottom frictional coefficient, respectively. Note that the partial operator in the matrix affects all of the factors to its right; e.g., the multiplication of $\frac{\partial\cos\varphi}{\cos\varphi\partial y}$ by V should be understood as $\frac{\partial(V\cos\varphi)}{\cos\varphi\partial y}$. To avoid the polar singularity, a rotated spherical coordinate system is used, the pole of which is rotated to (40 W, 80 N), which corresponds to a point on land in Greenland.

[1] A mass flux is defined as the product of the depth-averaged velocity and the water depth.

The second term of the right-hand side (RHS) of Eq. (1) contains the forces of the atmospheric pressures and wind stresses. The air pressures at the mean sea level, p_a, enter into the momentum equation as the inverse barometer η_a

$$\eta_a = -\frac{p_a}{\rho g} \tag{2}$$

where ρ is the density of seawater, which is taken as 1025 kg/m^3. The unit of η_a is meters. The wind stresses τ_x and τ_y are obtained by converting the wind velocity components, U_{10} and V_{10}, at 10 m above sea level with

$$(\tau_x, \tau_y) = \frac{\rho_a}{\rho} C_d \sqrt{({U_{10}}^2 + {V_{10}}^2)}\,(U_{10}, V_{10}) \tag{3}$$

where ρ_a refers to the air density, taken as 1.25 kg/m^3, and C_d is the drag coefficient, which is specified by

$$C_d = \begin{cases} 1.6\times10^{-3}, & \left(\sqrt{U_{10}^2+V_{10}^2}\le 7\ \mathrm{m\ s^{-1}}\right),\\ 2.8\times10^{-3}, & (\text{otherwise}).\end{cases} \tag{4}$$

The formula for the drag coefficient was adapted from Csanady (1982). The second line of Eq. (4) is a modification that was obtained by trial and error from fitting our model solutions to a real storm surge. The wind stresses defined by Eq. (3) are known as the kinematic stresses and are given in units of square meters per square second.

A linear frictional stress, $\kappa(U, V)$/h, is used at the sea bottom following Heaps (1969). However, Heaps used a constant κ=0.0024 m/s, whereas a spatially varying κ is adopted here following Ding et al. (2004) and Tan (1992) such that it is inversely proportional to the cubic root of the water depth. This produces κ values that range from 4.5×10^{-4} to 4.6×10^{-3} m/s across the world ocean. This approach is an attempt to reflect the lower values of bottom friction in deep water compared with shallow water; see Xu (2011) for additional details.

The world ocean is taken as the model domain, and GEBCO08 (General Bathymetric Chart of the Ocean; http://www.gebco.net) is used for the model's bathymetry. An advantage of using the global ocean as the model domain is that the model is free of artificial open-water boundaries; all of the lateral boundary conditions are zero normal flow conditions at the coasts:

$$U = 0 \quad \text{at the west and east coasts,} \tag{5}$$

$$V = 0 \quad \text{at the south and north coasts.} \tag{6}$$

Equation (1) contains differential operators in space and time. Xu (2011) detailed how to replace the differential operators with difference operators using the central difference in space and the explicit-implicit (EI) scheme of

Sielecki (1968) in time. This study used the same central difference scheme in space, but the alternating directon implicit (ADI) scheme of Leendertse (1967) is adopted for time. An advantage of using the ADI scheme is that its time step is not restricted by the CFL condition (Courant et al. 1967). Appendix 1 shows the ADI scheme in matrix form.

Regardless of the difference schemes, we can always arrive at the following canonical form:

$$\begin{bmatrix}\eta\\ U\\ V\end{bmatrix}^{(k+1)} = \mathbf{A}\begin{bmatrix}\eta\\ U\\ V\end{bmatrix}^{(k)} + \mathbf{B}\begin{bmatrix}\eta_a\\ \tau_x\\ \tau_y\end{bmatrix}^{(k)} \quad (k = 0, 1, 2, \cdots, k_{\max}) \tag{7}$$

where the bold letters are the discretized versions of their continuous counterparts in Eq. (1), k is the time-stepping index, and $k_{\max}$ is determined by $k_{\max}=fix(T/\Delta t)$, where T is the duration of a model run, Δt is the time step of the model, and *fix* is a function that rounds its argument to the nearest integer toward zero. In this paper, a superscript in parenthesis indicates a time-stepping index.

Matrix **A** updates the state vector $[\eta \;\; U \;\; V]^T$ from the current time step to the next; it has various names, including the dynamics matrix, the propagation matrix, and the updating matrix, depending on the context. Matrix **B** maps the atmospheric forcing into the momentum to change the state vector. By introducing **x** to denote the state vector and **f** to denote the forcing vector

$$\mathbf{x} \equiv [\,\eta \quad U \quad V\,]^T \tag{8}$$

$$\mathbf{f} \equiv \mathbf{B}[\,\eta_a \quad \tau_x \quad \tau_y\,]^T \tag{9}$$

we can present Eq. (7) in a compact form:

$$\mathbf{x}^{(k+1)} = \mathbf{A}\mathbf{x}^{(k)} + \mathbf{f}^{(k)}, \quad (k = 0, 1, 2, \cdots, k_{\max}). \tag{10}$$

Xu (2011) showed how the updating matrix **A** can be generated for Sielecki's (1968) EI difference scheme. Appendix 1 shows how the updating matrix **A** can be obtained as a product of four factor matrices for Leendertse's (1967) ADI difference scheme. One may substitute in their own favorite difference scheme. In this case, the contents of matrix **A** will change, but the form of Eq. (10) remains the same. Therefore, Eq. (10) can be viewed as a canonical form for representing all of the depth-averaged linear SWE models.

From Eq. (10) we can see that $\mathbf{x}^{(k+1)}$ has to be updated from $\mathbf{x}^{(k)}$. This means that even if we are interested in the solution to only one of the elements of the state vector, the solutions to all of the other elements must be computed. The solution to one of the elements may mean a time series of the sea surface elevations at a POI (in practice, we only have a few such POIs where we want model solutions). To obtain the time series at the POI, the solutions at all of the model grid points must be computed and then discarded, except for the one that is of interest. Thus, an enormous portion of the computations is wasted on the vast area of the ocean where solutions are not of interest. This has been the traditional way of modeling and the waste has been viewed as being necessary. However, we can actually avoid it using the ASGF method, as will be demonstrated in the next section.

The ASGF method uses Eq. (10) only to calculate a convolution matrix that contains all the Green's functions for a POI. Having prepared the convolution matrix, Eq. (10) will no longer be used. This is because the convolution matrix is an internal property of the ocean pertinent to the POI and only needs to be calculated once. Afterward, any storm surge simulation can be achieved through the convolution of the Green's function matrix with an atmospheric forcing field. All of the computations for the convolution are directly targeted toward the POI; no computation is wasted elsewhere. Thus enormous computational efficiencies can be gained.

The computational efficiency gain by the ASGF method compared with the traditional method will be theoretically and empirically studied in detail in Section 3 of part II, where Eq. (10) will be used to represent the traditional method because of its being canonical, because the ASGF is prepared with it, and because without the ASGF method, Eq. (10) would have to be used to model storm surges.

3 Storm surge solution and the ASGF

This section presents the definition of and algorithm for the ASGF and the solutions to the initial value and forced-wave problems. It then introduces the concepts of the memory time scale and the sampling rate for the ASGF. These two concepts are important for economically computing and storing the ASGFs; they will also be used in part II to assess the gain in computational efficiency obtained using the ASGF method.

3.1 The ASGF and its convolution

The solution to Eq. (10) can be expressed in terms of the initial condition and the external force field as

$$\mathbf{x}^{(k+1)} = \mathbf{A}^{k+1}\mathbf{x}^{(0)} + \sum_{i=0}^{k} \mathbf{A}^i \mathbf{f}^{(k-i)}, \quad (k = 0, 1, 2, \cdots, k_{\max}) \tag{11}$$

where the superscripts without parentheses refer to powers of the matrix. At first glance, the solution may appear to be impractical because it requires powers of the matrix **A**, and powers of a large matrix are computationally expensive. This would be the case if we needed to find the solutions at all the model grid points; however, we only need to know the

solutions at a few POIs. In this case, we only need to calculate a few rows of the matrix powers instead of all of them. Without loss of generality, let us assume that the solution is needed at only one POI corresponding to the nth grid point, i.e., only the nth component of the solution vector $\mathbf{x}$ is of interest. In this case, as shown in Eq. (11), only the nth row of each of the powers of $\mathbf{A}$ is needed. Introducing a new notation

$$\mathbf{r}_k \equiv \mathbf{A}^k(n,:) \tag{12}$$

to represent the nth row of the kth power of $\mathbf{A}$, we can then write

$$\eta^{(k+1)} = \mathbf{r}_{k+1}\mathbf{x}^{(0)} + \sum_{i=0}^{k} \mathbf{r}_i \mathbf{f}^{(k-i)}, \quad (0 \le k \le k_{\max}) \tag{13}$$

where $\eta \equiv \mathbf{x}(n)$ explicitly indicates that the nth component of the state vector is a sea surface elevation, which is of particular concern in this paper (but the velocities at a POI can be of interest as well). The row vector $\mathbf{r}_i$ can be iteratively calculated as

$$\mathbf{r}_{k+1} = \mathbf{r}_k \mathbf{A}, \quad \text{for } 0 \le k \le k_{\max} \tag{14}$$

$$\mathbf{r}_0(j) = \begin{cases} 0, & j \text{ for all the grid points except for the } n\text{th}, \\ 1, & j = n\,. \end{cases} \tag{15}$$

Each iteration involves a multiplication of a row vector by a matrix, which can be performed very economically. Appendix 2 shows how to calculate the row vector $\mathbf{r}_i$ when the matrix $\mathbf{A}$ is given in its factor matrices. By collecting all of the row vectors into two matrices,

$$\mathbf{G_i} = [\mathbf{r}_1; \mathbf{r}_2; \cdots; \mathbf{r}_{k_{\max}}; \mathbf{r}_{k_{\max}+1}], \tag{16}$$

$$\mathbf{G_c} = [\mathbf{r}_0; \mathbf{r}_1; \mathbf{r}_2; \cdots; \mathbf{r}_{k_{\max}}], \tag{17}$$

where the semicolon ";" indicates the end of the preceding row and the beginning of the next row (i.e., the $\mathbf{r}$s are vertically stacked), we can concisely express Eq. (13) as

$$\boldsymbol{\eta} = \mathbf{G_i}\mathbf{x}^{(0)} + \mathbf{G_c} * \mathbf{f} \tag{18}$$

where $\boldsymbol{\eta} = \left[\eta^{(1)}\eta^{(2)}\eta^{(3)}\cdots\eta^{(k_{\max}+1)}\right]^T$ is a column vector that contains the time series of the solution for the sea surface elevations at the POI. The second term is a convolution and is defined as

$$(\mathbf{G_c} * \mathbf{f})^{(k+1)} \equiv \sum_{i=0}^{k} \mathbf{G_c}(i+1,:)\,\mathbf{f}^{(k-i)}, \quad (k = 0, 1, 2, \cdots, k_{\max}) \tag{19}$$

where the notation "*" represents the convolution operation. Equation (18) shows that the solution is composed of two parts: the first term of the RHS is the contribution of the initial condition, and the second term is the contribution of the external forcing. The forcing vector $\mathbf{f}$ changes with time. A convolution is needed because different instances of $\mathbf{f}$ produce different responses, which must be added in the correct order in time.

As shown in Eqs. (16) and (17), the same set of $\mathbf{r}$-vectors appears in both $\mathbf{G}_i$ and $\mathbf{G}_c$, except $\mathbf{r}_0$ only appears in $\mathbf{G}_i$ and $\mathbf{r}_{k_{\max}+1}$ only appears in $\mathbf{G}_c$. The matrix $\mathbf{G}_i$ is appropriate for free-wave problems such as tsunami propagation, whereas the matrix $\mathbf{G}_c$ is suitable for forced-wave problems such as storm surges. Either matrix can be used as the definition of the ASGF. In this paper, when there is no ambiguity, their subscripts may be omitted. The ASGF is an internal property of the dynamic system; it can be pre-calculated. Once it has been calculated, it can be repeatedly used to rapidly produce the response to any event such as a tsunami or a storm surge.

The columns of the $\mathbf{G}$ matrix contain the Green's functions corresponding to all of the model grid points, one column for one Green's function to an impulse at a grid point. The algorithm shown in Eqs. (14) and (15) is a new way to calculate the Green's functions. It completely eliminates the source limitation problem associated with the traditional way to calculate the Green's functions.

Not all of the Green's functions contained in the columns of the $\mathbf{G}$ matrix share the same unit. The units also depend on what the variable is interested at the POI for which the $\mathbf{G}$ matrix is calculated. If the sea surface elevations are the variable of interest at the POI, the Green's functions contained in the columns of $\mathbf{G}$ are either nondimensional or have a dimension of square seconds per meter; the nondimensional functions are contained in the columns corresponding to the atmospheric pressures expressed as the inverse barometer, η_a, and the dimensional functions are contained in the columns corresponding to the wind stresses, τ_x and τ_y. This can be deduced from Eq. (1) by noting that the inverse barometer pressure, η_a, has units of meters and that the kinematic wind stresses, τ_x and τ_y, have units of square meters per square second.

For tsunami propagation in the ocean, which is a free-wave problem because there is no external forcing after the onset of a tsunami, Eq. (18) reduces to

$$\boldsymbol{\eta} = \mathbf{G_i}\mathbf{x}^{(0)}. \tag{20}$$

A matrix times a column vector can be rapidly calculated. This means that we can *instantaneously* produce a tsunami arrival time series at a destination point, if a reliable initial condition $\mathbf{x}^{(0)}$ can be also made available at the same time. In the tsunami literature, an initial condition is called a source function. Xu

and Song (2013) demonstrated the potential for rapid tsunami prediction by combining the ASGF method and the GPS-derived source function (based on the ground movements of the coastal GPS stations that were detected by satellites) using the 2011 Tohoku tsunami as an example.

For a storm surge problem, the forcing field **f** exists, occupies the entire model domain, and changes with time. After the forcing spins up the ocean, the convolution term becomes dominant, whereas the effects of the initial condition become negligible due to friction. Therefore, we can drop the initial condition term and simplify Eq. (18) as

$$\boldsymbol{\eta} = \mathbf{G}_c * \mathbf{f}. \tag{21}$$

The forcing vector **f** is specified using the outputs from an atmospheric model. Usually, an atmospheric model has a coarser spatial resolution than does a surge model in the ocean. In this case, the forcing vector and the columns of $\mathbf{G}_c$ can be compressed to reduce their sizes and hence to enhance the computational and storage efficiencies. Appendix 3 discusses this point in detail.

3.2 The memory time scale of the ocean and the length of the convolution kernel

Equation (14) appears to suggest that the row vectors, $\mathbf{r}_{k+1},\quad (k = 0, 1, 2, \cdots, k_{\max})$, need to be iterated up to $k=k_{\max}$. This is not necessary in an energy dissipative system when $k_{\max}$ is large. Due to friction, the magnitudes of all of the elements in the row vectors will gradually become negligibly small. The real ocean has certain memory time scales to remember certain things. We may divide a row vector in the **G** matrix into two sub-row vectors, one corresponding to the air pressures and another corresponding to the wind stresses. Figure 1 shows the attenuation of the sub-row vectors with time; the top panel shows the attenuation of the infinite norm of the sub-row vectors corresponding to the air pressures, and the bottom panel shows the attention of the infinite norms of the sub-row vectors corresponding to the wind stresses. As we can see, the infinite norms already become negligible after 48 h. To be conservative, this study chose 72 h as the memory time scale of the ocean, T_{mem}, after which we can replace any **r**-vector by a zero vector. The memory time scale T_{mem} may also be called the convolution kernel length because it dictates how long the kernel should extend to the past, as we will see soon.

The fact that the ASGF attenuates with time is an advantage that we should take to reduce the computational load and storage space. By setting an appropriate memory time scale, T_{mem}, we do not need to calculate the **r**-vectors after T_{mem}. This consideration leads to a split of the recursion scheme in Eq. (14) into two parts:

$$\mathbf{r}_{k+1} = \mathbf{r}_k \mathbf{A}, \quad (0 \le k \le k_{\text{mem}}) \tag{22}$$

$$\mathbf{r}_{k+1} \approx \mathbf{0}, \quad (k_{\text{mem}} + 1 \le k \le k_{\max}) \tag{23}$$

where $k_{\text{mem}}=\text{fix}(T_{\text{mem}}/\Delta t)$. Instead of stopping at $k=k_{\max}$, the recursion now stops at $k=k_{\text{mem}}$, after which any $\mathbf{r}_{k+1}$ can be simply approximated by a zero vector. Accordingly, the $k_{\max}$ in Eqs. (16) and (17) should be replaced by k_{mem},

$$\mathbf{G_i} = [\mathbf{r}_1; \mathbf{r}_2; \cdots; \mathbf{r}_{k_{\text{mem}}}; \mathbf{r}_{k_{\text{mem}}+1}], \tag{24}$$

$$\mathbf{G_c} = [\mathbf{r}_0; \mathbf{r}_1; \mathbf{r}_2; \cdots; \mathbf{r}_{k_{\text{mem}}}], \tag{25}$$

and Eq. (13) should also be split into two parts:

$$\eta^{(k+1)} = \mathbf{r}_{k+1}\mathbf{x}^{(0)} + \sum_{i=0}^{k} \mathbf{r}_i \mathbf{f}^{(k-i)}, \quad (0 \le k \le k_{\text{mem}}) \tag{26}$$

$$\eta^{(k+1)} \approx \sum_{i=0}^{k_{\text{mem}}} \mathbf{r}_i \mathbf{f}^{(k-i)}, \quad (k_{\text{mem}} + 1 \le k \le k_{\max}). \tag{27}$$

Equation (26) shows that within the memory time scale, the sea level at the next time step, $\eta^{(k+1)}$, is affected by the initial condition and by a sum of weighted forcing vectors of the present time step, $\mathbf{f}^{(k)}$, and of all of the past time steps, $\mathbf{f}^{(k-1)}, \mathbf{f}^{(k-2)}, \cdots, \mathbf{f}^{(0)}$. Equation (27) shows that beyond the memory time scale, the initial condition no longer has any effect; the sea level at the next time step, $\eta^{(k+1)}$, is purely a sum of the weighted forcing vectors of the present, $\mathbf{f}^{(k)}$, and of the past, $\mathbf{f}^{(k-1)}, \mathbf{f}^{(k-2)}, \cdots, \mathbf{f}^{(k-k_{\text{mem}})}$, with $\mathbf{r}^{(0)}$ as the weights of the current forcing vector $\mathbf{f}^{(k)}$, $\mathbf{r}^{(1)}$ as the weights of the immediate past forcing vector $\mathbf{f}^{(k-1)}$, etc. The weights $\mathbf{r}^{(k)}$ diminish with k and are simply replaced by zero vectors for any k larger than k_{mem}, which means that any forcing vector in the past before $t=k_{\text{mem}}\times\Delta t$ has no effect on the sea level at the next time step. Continuation of this weighted sum process as k increases is what a convolution is all about. The **r**-vectors form a kernel of the convolution, with $\mathbf{r}^{(0)}$ and $\mathbf{r}^{(k_{\text{mem}})}$ as the head and tail of the kernel, respectively, and k_{mem} is the length of the kernel.

Equation (26) is an exact relation, which means that the solution to the sea surface elevation at a point, $\eta^{(k+1)}$, obtained by this equation will be identical to the solution obtained by the traditional method, i.e. Eq. (10), for $k \le k_{\text{mem}}$. Equation (27) is an approximate relation, which means that the solution obtained by Eq. (27) will not be identical to but approximate the solution at the same point that is obtained by Eq. (10) for $k > k_{\text{mem}}$. The error is due to the truncation of the **r**-vectors shown in Eq. (23). However, the truncation error can be controlled by choosing an appropriate value for k_{mem} to make the solution obtained by Eq. (27) *practically the same* as that obtained by Eq. (10). Saying that two solutions are *practically the same* it is meant that their differences are insignificant for all practical purposes.

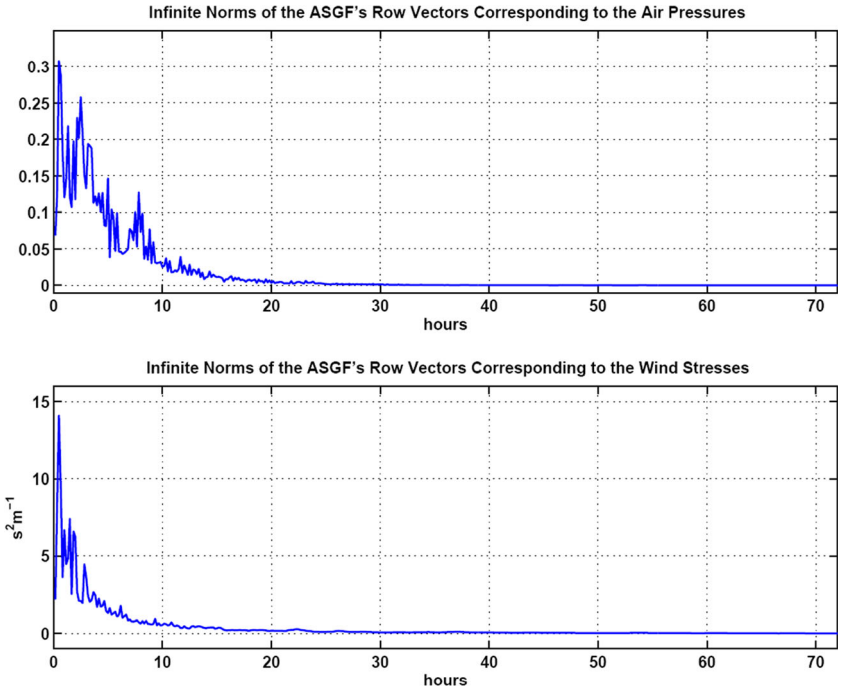

Fig. 1 Attenuation of the ASGF with time. *Top*, attenuation of the infinite-norm of the sub-row vectors of the matrix corresponding to the air pressures. *Bottom*, attenuation of the infinite-norm of the sub-row vectors of the matrix corresponding to the wind stresses

3.3 The sampling rate for the ASGF

With the concept of the memory time scale introduced above, the $\mathbf{G}_c$ (or $\mathbf{G}_i$) matrix should now be viewed as a collection of row vectors $\mathbf{r}_{k+1}$ up to k_{mem} (or $k_{\text{mem}}+1$, cf. Eqs. (16) and (17)). When $k_{\text{mem}} \ll k_{\max}$, the number of the rows in the $\mathbf{G}$ (either $\mathbf{G}_i$ or $\mathbf{G}_c$) matrix is greatly reduced. We can further reduce the number of rows if we find that the Δt associated with the k-index is too fine for the problem in question. In this case, we can sample only a subset of the $\mathbf{r}$-vectors. For example, if $\Delta t = 5$ sec, which may be imposed by the CFL condition for stability, and if we collect all of the row vectors in the $\mathbf{G}$ matrix, the time series of η on the left-hand side (LHS) of Eqs. (26) and (27) would also have a 5-sec time resolution, which might be excessive for many practical problems. For example, a 1-min resolution would be adequate for a tsunami arrival time series at a POI; in modeling storm surges, a 1-hour resolution is commonly used in outputting the model solutions. We could then sample the $\mathbf{r}$-vectors at a decimating rate of 12 or 720 for the tsunami or storm surge problems, respectively. In other words, the sampling rate for the $\mathbf{G}$ matrix can be expressed as

$$\Delta t_{\text{smp}} = d\Delta t, \quad (d \geq 1) \tag{28}$$

where d represents the decimating rate. The sampling rate for the ASGF can be minutely, hourly, or any other time interval depending on the requirements of the problem. Note that regardless of how Δt_{smp} is chosen, the accuracy of the $\mathbf{r}$-vectors will not be affected. The $\mathbf{r}$-vectors are the results of the iterations of Eqs. (22) and (23); their accuracies are already fixed when Δt is chosen in assembling the updating matrix $\mathbf{A}$. The choice of Δt_{smp} only matters how frequently we sample the $\mathbf{r}$-vectors.

In choosing a decimating rate for storm surge modeling, we have to also consider the time resolution in a given atmospheric forcing field. If the given field has an hourly resolution, the decimating rate d should be chosen such that $\Delta t_{\text{smp}} = d\Delta t = 3600$ sec. If the given atmospheric forcing field has a 3-hour resolution (which is not uncommon), the decimating rate d may be chosen as $\Delta t_{\text{smp}} = d\Delta t = 3 \times 3600$ sec.

The $\mathbf{r}$-vectors are produced with a finer time resolution, Δt, but are sampled with a coarser one, Δt_{smp}. During each Δt_{smp}, the forcing is assumed to be constant. The accumulative effect of the constant forcing during the d steps of Δt is considered in Appendix 4. The appendix also presents two MATLAB functions, ASGF_ini and ASGF_conv, that are used to calculate the $\mathbf{r}$-vectors and to assemble them into the $\mathbf{G}_i$ and $\mathbf{G}_c$ matrices for the initial value problem and for the forced problem. In the functions, we can also see how the decimating rate, d, is used.

Based on Eqs. (24) and (25) and the sampling rate introduced above, we can see that the number of rows of the matrix $\mathbf{G}$, denoted by L_G, is

$$L_G = fix\left(T_{\text{mem}}/\Delta t_{\text{smp}}\right). \tag{29}$$

Because the matrix $\mathbf{G}$ plays the role of the convolution kernel, its number of rows may also be referred to as the length of the convolution kernel or simply the length of $\mathbf{G}$.

4 Interpretations of the ASGF

This section interprets the ASGF from different perspectives and links it to several familiar concepts.

4.1 Dependence field

The matrix **G** can be interpreted with physical meanings. Figure 2 should remind us of a familiar concept often seen in text books: the dependence intervals for a one-dimensional wave solution at a point. The figure shows that the wave solution at a point of interest, x, depends only on the conditions within the interval $(x-ct,x+ct)$, where c is the wave speed, which is constant in this case. The dependence interval increases over time at the same rate as the wave speed c.

The wave solution at a point on the real ocean surface also has a domain of dependence. However, this seems to have received little attention in practice, perhaps because it is difficult to visualize this domain from solutions that are obtained with conventional modeling approaches. Now, from the **G** matrix, we can not only see the domain of dependence but also know the weights of the dependence. Each row of **G** contains the domain of dependence at a particular time. Figure 3 shows the domains of dependence of the wave solution at Sept-Îles (Quebec, Canada) at four different times. Figure 3a shows the domain of dependence at $t=6$ hour; the solution at Sept-Îles depends only on the conditions within the colored region. The conditions outside the region do not yet affect the solutions; they need more time to affect the solution. From Fig. 3a–d, we can see how the domain of dependence grows.

Their growth rates are controlled by the wave speeds, which in turn are controlled by the water depths varying spatially in the real ocean. As shown in Fig. 3d, the domain of dependence covers almost the entire world ocean in 48 hours, which means that anything that occurs in the world ocean can significantly or insignificantly affect the wave solution at Sept-Îles within 2 days. The color spectrum indicates the weights of the dependences, which can be positive or negative. A negative weight means that a positive impulse will cause a negative response. We can collectively refer to the domains and weights of dependence as a field of dependence. The values of the weights are largely affected by the resolution of the model grid; the finer the grid spacing is, the smaller the weights will be; however what matters is the spatial integral of the weights. The weights shown in Fig. 3 are for a model grid spacing of 5 min in longitude and latitude. The field of dependence may also be called the connectivity between the POI and the rest of the world ocean.

The columns of **G** contain all the Green's functions; each column is a response time series to an impulse placed at a grid point. There are as many such Green's functions as there are grid points. The name "all-source Green's function" means that all of the model grid points can be source points. Thus **G** contains a complete set of information about the linear dynamic system. Its rows contain information about how the POI is connected to the rest of the world (i.e., the fields of dependence), and its columns contain all the Green's functions to the delta forcings at all the grid points. We may also say that its columns contain all the temporal information and its rows contain all the spatial information. The matrix **G**, or the ASGF, is an internal property of the linear dynamics system of the world ocean. It is independent of external forcings and can be calculated before events occur. Once it is pre-calculated, it can be repeatedly used to quickly calculate responses to tsunami and storm surge events. It can also be used to model tides (which will be the topic of another paper). All of the time-consuming computations (such as those due to the small time steps) have been absorbed into the calculation of **G**.

4.2 An MISO system

In the language of system and control theory, the ASGF is a system of multiple inputs and a single output (MISO, Fig. 4). The multiple inputs are a global forcing field, **f**, that is defined at model grid points, and the single output is a response time series, **η**, at a POI. The ASGF, which is the matrix **G**, is the kernel of the MISO system. With the same **G** but a different type of forcing field **f**, the system becomes a different model. When **f** represents an atmospheric forcing field, the system is a storm surge model; when **f** represents an astronomical forcing

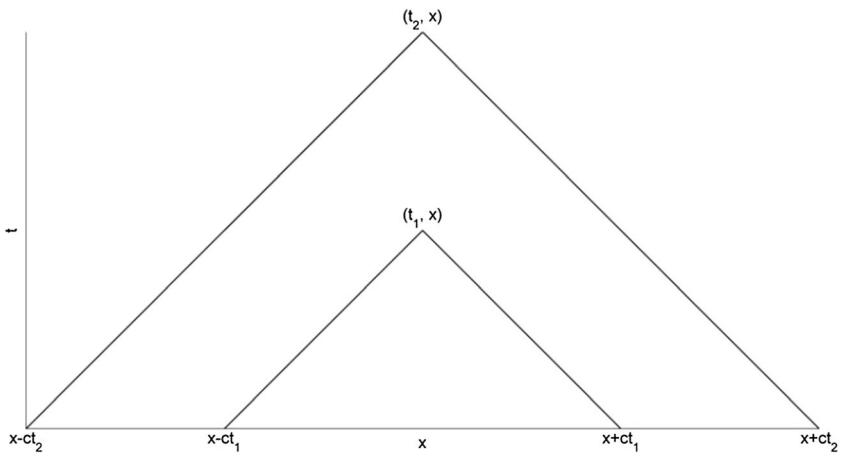

Fig. 2 The one-dimensional wave solution at point x depends on the initial conditions only within the interval $(x-ct, x+ct)$. The interval of dependence grows at the same rate as the wave speed c

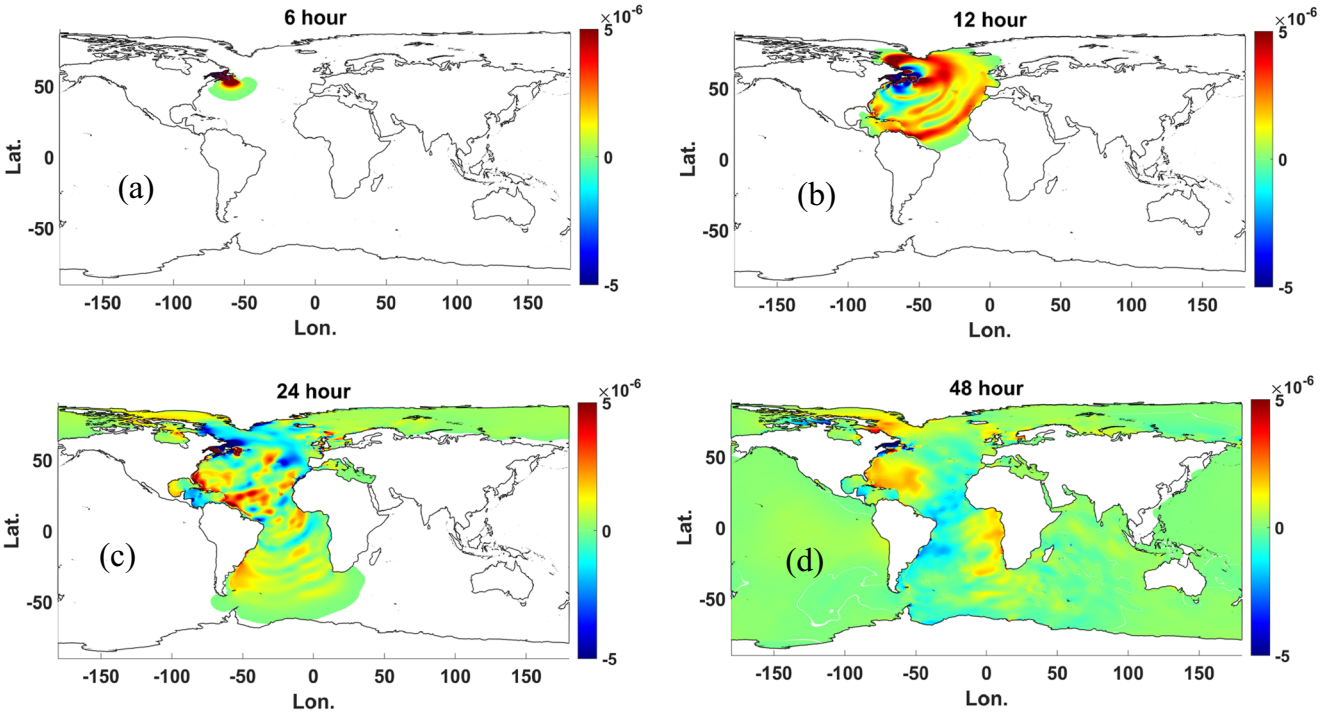

Fig. 3 Domains of dependence of the wave solutions at Sept-Îles, Gulf of St. Lawrence, at t=6, 12, 24, and 48 hour (**a**–**d**). The *colors* indicate the weights of the dependences. Note that the longitudes and latitudes are rotated longitudes and latitudes that are used by the model. The pole of the spherical coordinates that is used by the model is set at a location in Greenland (40 W, 80 N) to avoid having the polar singularity located in the water

field, the system is a tide model; and when **f** is tectonic (via a source function), the system is a tsunami propagation model (Fig. 4).

5 Summary and discussion

Beginning with the depth-averaged linear shallow-water equations, this paper defined the ASGF and presented its algorithm for two situations: one where the dynamics matrix **A** is available as a single matrix, which results from using a simple numerical discretization scheme, such as Sielecki's (1968) EI scheme, and one where the dynamic matrix **A** can only be presented in factor matrices. The latter situation arises from the use of a more advanced discretization scheme such as the ADI. Appendixes 1 and 2 present the ADI scheme in matrix form and the corresponding ASGF algorithm, respectively. In addition to the definition of and algorithm for the ASGF, the

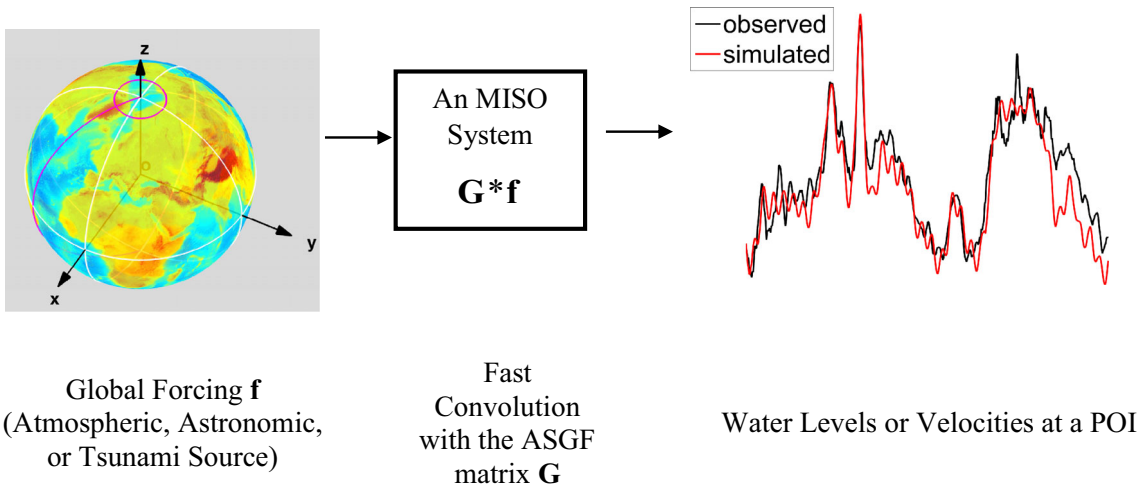

Fig. 4 An MISO system with the ASGF as its kernel. The global forcing field can be atmospheric, astronomical, or tectonic and can be defined over the entire domain or at any part of the domain. A convolution of the ASGF matrix **G** with the forcing field can quickly yield a response

concepts of the memory time scale and the sampling rate of the ASGF were also introduced. The two concepts make the ASGF computed and stored economically. It will be shown in part II that they are also important factors to enhance the computational efficiency of the ASGF method in comparison with the traditional method.

The rows and columns of the ASGF have different meanings. The rows of the ASGF matrix contain information about how the POI is connected to the rest of the world ocean at different times (more precisely called the field of dependence), and the columns of the matrix contain all of the Green's functions that correspond to the impulse forces at all of the grid points. In the terminology of system and control theory, the ASGF was also interpreted as a system of multiple inputs and a single output (MISO).

Equation (10) is a canonical form to represent various traditional linear storm surge models, although they may not be written in matrix form; different models differ only in the content of the updating matrix **A**. Two features are common to all the traditional storm surge models: they all must map out solutions at every grid point, even though only solutions at a few grid points are of interest, and they all have to use small time steps to ensure stability or accuracy, even though hourly outputs are commonly needed. These two features result in intensive computations. Consequently, global storm surge models are rare; most are regional. A regional model has a lower computational load, but the trade-off is the challenge of the artificial open-water boundary conditions.

The ASGF method that is proposed in this paper improves upon the traditional modeling approach. Instead of being run for individual events, Eq. (10) is used only to calculate the ASGF. The ASGF needs to be calculated once; afterward it can be repeatedly used for any event. A convolution of the ASGF matrix with a forcing field will quickly yield the response to an event. The ASGF also accounts for the influences of global forcing fields and the global ocean geometry. It bypasses the open-water boundary condition issue because it can efficiently include the entire world ocean as its domain.

The ASGF simplifies the expression of a storm surge model. It expresses the sea surface elevations at a point as a convolution of the ASGF matrix with a forcing field. This simple expression opens a door to many other mathematical operations, such as singular value decomposition (SVD), the fast Fourier transform (FFT), and linear regression analyses, which can make storm surge modeling even faster and data-assimilative. These points will be considered in part II of this study. The mathematical formulas that were developed in this paper will also be tested through both simplified and realistic cases in part II. The formulas will be validated with an analytical solution in the simplified case and will be tested with field observational data in the realistic case.

Acknowledgments This study was partially supported by the Ministère des Transports du Québec and by the ACCASP program of Fisheries and Oceans. This study also benefited from collaborations with the OURANOS Consortium (http://www.ouranos.ca/). Free access to the MERRA data by NASA is greatly appreciated. The author also gratefully acknowledges support from his own institute.

Appendixes

Appendix 1: the ADI scheme in matrix form

The ADI scheme for modeling long waves in the ocean was first developed by Leendertse (1967). This appendix presents the ADI scheme in a form of a product of factor matrices and shows that the form of Eq. (10) still holds.

It is always possible to turn Eq. (1) into a semi-discretized form as

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{C}\mathbf{x} + \mathbf{D}\hat{\mathbf{f}}, \tag{30}$$

where **x** is the same as was defined by Eq. (8), the matrices **C** and **D** host the spatial difference operators, and

$$\hat{\mathbf{f}} \equiv \begin{bmatrix} \eta_{\mathbf{a}} & \tau_{\mathbf{x}} & \tau_{\mathbf{y}} \end{bmatrix}^{\mathrm{T}}. \tag{31}$$

This form is called a semi-discretized form because only the space is discretized, and the time remains continuous. It is easy to calculate **C** because only the spatial difference operators must be assembled. In addition, **C** is highly sparse. It is thus feasible to store it in a computer's RAM even for a fairly large system, such as the system used in this study, which contains 32,377,503 gridded variables from the 5-min discretization of the global ocean; its sparsity is on the order of 10^{-7}.

Various numerical schemes can arises when it comes to discretize the time derivative in Eq. (30). The ADI discretizes the time by first splitting the matrix **C** into two parts

$$\mathbf{C} = \mathbf{C}_{\mathbf{x}} + \mathbf{C}_{\mathbf{y}} \tag{32}$$

where $\mathbf{C}_{\mathbf{x}}$ only involves the *x*-directional spatial operators and $\mathbf{C}_y$ only involves the *y*-directional spatial operators. It then splits the time step into two half-time steps such that

$$\frac{\mathbf{x}^{(k+1/2)} - \mathbf{x}^{(k)}}{\Delta t/2} = \mathbf{C}_{\mathbf{x}}\mathbf{x}^{(k+1/2)} + \mathbf{C}_{\mathbf{y}}\mathbf{x}^{(k)} + \mathbf{D}\hat{\mathbf{f}}^{(k)}, \tag{33}$$

$$\frac{\mathbf{x}^{(k+1)} - \mathbf{x}^{(k+1/2)}}{\Delta t/2} = \mathbf{C}_{\mathbf{x}}\mathbf{x}^{(k+1/2)} + \mathbf{C}_{\mathbf{y}}\mathbf{x}^{(k+1)} + \mathbf{D}\hat{\mathbf{f}}^{(k+1/2)}. \tag{34}$$

In the first half-time step, the implicit scheme is applied only to the *x*-direction; in the second half-time step, the implicit scheme is applied only to the *y*-direction. Thus, only a tridiagonal matrix equation needs to be solved in each half step,

which is not costly. It is reasonable to assume that $\mathbf{f}^{(k+1/2)}=\mathbf{f}^{(k)}$. From Eqs. (33) and (34), we have

$$(\mathbf{I}-s\mathbf{C}_\mathbf{x})\mathbf{x}^{(k+1/2)} = \left(\mathbf{I}+s\mathbf{C}_\mathbf{y}\right)\mathbf{x}^{(k)} + s\mathbf{D}\hat{\mathbf{f}}^{(k)} \tag{35}$$

$$\left(\mathbf{I}-s\mathbf{C}_\mathbf{y}\right)\mathbf{x}^{(k+1)} = (\mathbf{I}+s\mathbf{C}_\mathbf{x})\mathbf{x}^{(k+1/2)} + s\mathbf{D}\hat{\mathbf{f}}^{(k)} \tag{36}$$

where $s=\Delta t/2$. By merging the two half-time steps, we have

$$\mathbf{x}^{(k+1)} = \mathbf{A}\mathbf{x}^{(k)} + \mathbf{f}^{(k)} \tag{37}$$

where

$$\mathbf{A} = \left(\mathbf{I}-s\mathbf{C}_\mathbf{y}\right)^{-1}(\mathbf{I}+s\mathbf{C}_\mathbf{x})(\mathbf{I}-s\mathbf{C}_\mathbf{x})^{-1}\left(\mathbf{I}+s\mathbf{C}_\mathbf{y}\right) \tag{38}$$

$$\mathbf{f}^{(k)} = \Delta t\left(\mathbf{I}-s\mathbf{C}_\mathbf{y}\right)^{-1}(\mathbf{I}-s\mathbf{C}_\mathbf{x})^{-1}\mathbf{D}\hat{\mathbf{f}}^{(k)}. \tag{39}$$

Equation (37) recovers the canonical form of Eq. (10). The ADI scheme is a two-step scheme. The derivation presented above may serve as an example of how to derive the canonical form if a different multiple step scheme is used.

Matrix $\mathbf{A}$ is given in four factor matrices. However, these factor matrices should not be multiplied out when the system is large for two reasons: performing the multiplications would be excessively time consuming, and the product matrix $\mathbf{A}$ would no longer be sufficiently sparse to be stored in a computer's RAM. To calculate $\mathbf{A}\mathbf{x}^{(k)}$, we should successively perform the multiplications of the factor matrices with the column vector from right to left; each of the multiplications results in a column vector.

The ADI scheme has two advantages: it is stable without restriction of the CFL condition, and it almost perfectly conserves the total mechanical energy of the system. Leendertse (1967) and Wesseling (2009) showed that the ADI method does not impose a necessary stability condition on the time step for a linear system similar to Eq. (37), based on the Fourier stability analysis (von Newman stability analysis). Although the Fourier stability analysis does not provide a sufficient condition for stability, a test run of model Eq. (37) for 10 days shows that the model is stable even when using Δt=600 sec and a 5-min longitude and latitude resolution in a gridded global ocean with realistic bathymetry as the model domain. In contrast, using the EI scheme of Sielecki (1968), the time step would necessarily be bounded by the CFL condition at 5 sec for the same global model domain.

The other advantage of the ADI scheme is that it almost perfectly conserves the total mechanical energy of the system. The total mechanical energy, which is the sum of the potential and kinetic energies in the domain, should be conserved in a nondissipative system. However, this physical law may be violated to various degrees by different schemes. An unstable scheme will result in an unbounded increase in the total energy. Although a stable scheme must necessarily keep the total energy bounded, such a scheme may cause the total energy to fluctuate around its initial value or decay. Having the total energy decay with time, which is known as numerical dissipation, is not good either. A good scheme should avoid the numerical dissipation and minimize the fluctuation of the total energy. Figure 5 compares the total energy conservation by two schemes for a relaxation problem, where an initial sea surface distribution is suddenly released in a frictionless water body (Fig. 6). The top panel of Fig. 5 shows how the ADI scheme keeps the total mechanical energy almost constant; this is indicated by the blue line, which appears to be flat in the chosen axes. The panel also shows that the potential energy (in green) and the kinetic energy (in red) vary with time; however, their sum, which is the total energy (in blue), remains constant. The middle panel shows the energy time series obtained using Sielecki's scheme. The total energy does not appear as flat in the top panel and exhibits noticeable fluctuations. The bottom panel provides a zoomed in view of the total energy variations when using the two schemes. The total energy from the ADI scheme still appears flat, whereas that from Sielecki's scheme exhibits large fluctuations. Actually the total energy fluctuates with both schemes, but the fluctuation amplitudes are 0.01 % of the initial total energy with the ADI scheme and 2.90 % with Sielecki's scheme. If the Crank-Nicolson scheme was employed here, the total energy would be perfectly conserved (see Durran 1999, p. 158). However, the Crank-Nicolson scheme is computationally too expensive to apply. The involved matrix inversion is not as easy to compute as the inversions of the lower and upper triangular matrices involved in the ADI method. The ADI scheme balances computational accuracy and efficiency and was therefore adopted in this study.

In developing the numerical model for this study, a convenient conversion was designed between the spherical and Cartesian coordinate systems to easily construct the matrices that host the spatial coordinate information in either coordinate system. The ADI scheme that is presented in Appendix 1 is applicable to both coordinate systems because the C and D matrices in Eq. (30) are the only place where the spatial coordinate information resides.

Appendix 2: algorithm for computing the ASGF with the ADI scheme

The algorithm given by Eqs. (14) and (15) must be adapted for the ADI scheme. The ADI scheme results in the matrix $\mathbf{A}$, which is composed of 4-factor matrices, as shown by Eq. (38). As previously noted, we should not multiply out the product of the factor matrices. In addition, there are matrix

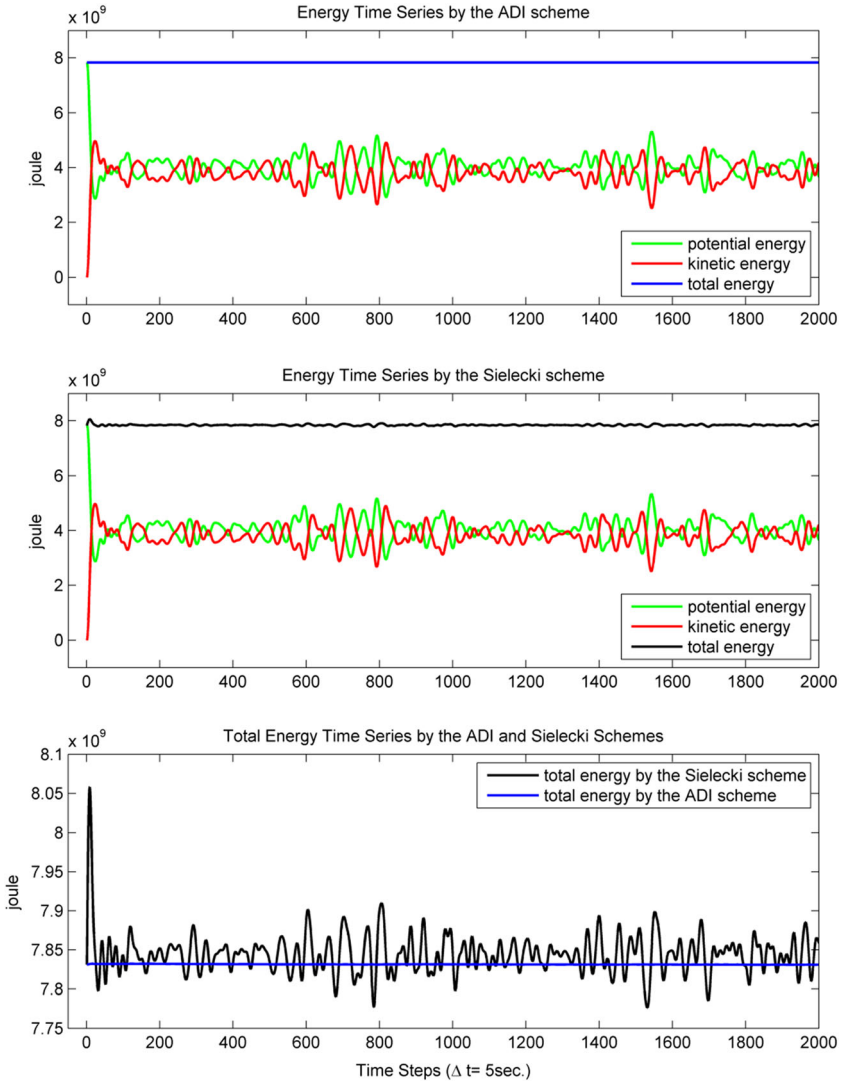

Fig. 5 Comparison of the conservation of total energy by the ADI and Sielecki's EI schemes for the relaxation problem (cf. Fig. 6). *Top* and *middle*, variations in the potential and kinetic energies and the total energies with time for the ADI and Sielecki schemes, respectively. *Bottom*, a zoomed in view of the differences in the fluctuations in the total energies of the two schemes. The ADI scheme conserves the total energy much better than does Sielecki's scheme

inverses among the factors, it becomes impossible to multiply a left row vector with a right matrix; the matrices must first be transposed. Transposing both sides of Eqs. (14) and (15), we have

$$\mathbf{c}_{j+1} = \left(\mathbf{I} + \frac{\Delta t}{2}\mathbf{C}_y\right)^{\mathrm{T}} \left(\left(\mathbf{I} - \frac{\Delta t}{2}\mathbf{C}_x\right)^{-\mathrm{T}} \left(\left(\mathbf{I} + \frac{\Delta t}{2}\mathbf{C}_x\right)^{\mathrm{T}} \left(\left(\mathbf{I} - \frac{\Delta \mathbf{t}}{2}\mathbf{C}_y\right)^{-\mathrm{T}} \mathbf{c}_j\right)\right)\right) \tag{40}$$

$$\mathbf{c}_0(i) = \begin{cases} 0, & \text{i for all the grid points except for the } n\text{th} \\ 1, & i = n \end{cases} \tag{41}$$

where $j=0,1,2,\cdots,j_{\max}$,

$$\mathbf{c} \equiv \mathbf{r}^{\mathrm{T}} \tag{42}$$

and $(\cdots)^{-\mathrm{T}}$ is short for $((\cdots)^{\mathrm{T}})^{-1}$. Once the $\mathbf{c}_{j+1}(j=0,1,2,\cdots, j_{\max})$ are calculated, we can transpose them back into the row vectors $\mathbf{r}_{i+1}(i=0,1,2,\cdots,i_{\max})$, which then can be stacked to construct the ASGF matrices, $\mathbf{G}_i$ and $\mathbf{G}_c$, as shown by Eqs. (24) and (25).

Appendix 3: reduction of the columns of the ASGF matrix

It is commonly seen that grid spacing in an atmospheric model is much coarser than in an oceanic model. For example, the spatial resolution of the global ocean surge model used in this study is 5-min in longitudes and latitudes. The air pressures and winds output from a global atmospheric model called MERRA (http://gmao.gsfc.nasa.gov/merra/) were used to specify the forcing vector. The spatial resolution of the MERRA model is 0.5 degrees in latitude and 0.67 degrees in longitude. This appendix considers how the coarser atmospheric forcing spatial resolution can be used to reduce the number of columns of $\mathbf{G}_c$. Equation (9) defines the forcing vector $\mathbf{f}$ as

$$\mathbf{f} \equiv \mathbf{B}[\,\eta_a \quad \tau_x \quad \tau_y\,]^{\mathrm{T}}, \tag{43}$$

where the matrix $\mathbf{B}$ maps the vector containing the air pressures η_a and wind stresses τ_x and τ_y onto $\mathbf{f}$. The air

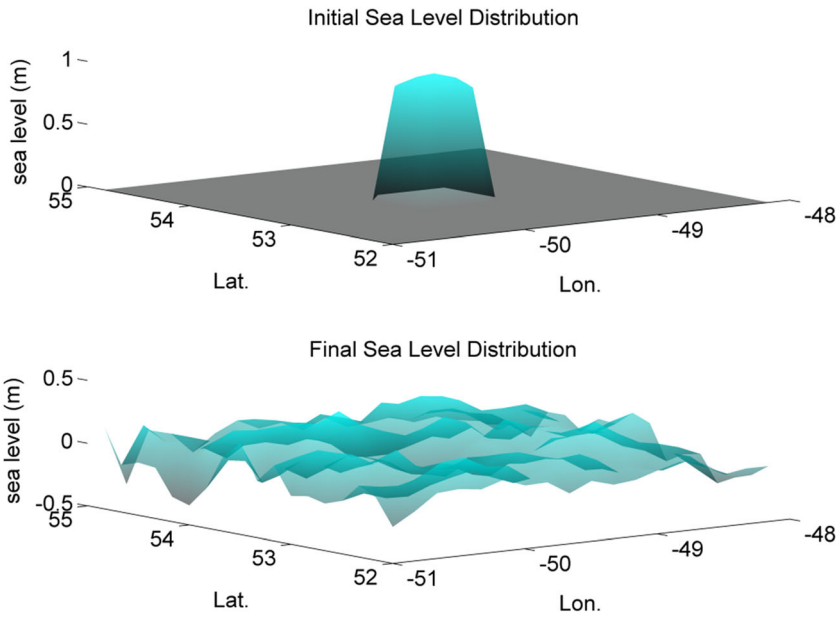

Fig. 6 A relaxation experiment. *Top*, the initial sea-level distribution; *bottom*, the distribution at time step 2000 with Δt=5 sec

pressures and wind stresses are interpolated from an atmospheric model grid, i.e.,

$$[\eta_a \quad \tau_x \quad \tau_y]^T = \mathbf{L}\left[\widetilde{\eta_a} \quad \widetilde{\tau_x} \quad \widetilde{\tau_y}\right]^T \tag{44}$$

where the variables with tildes are defined on an atmospheric model grid, those without are defined on the ocean model, and **L** is the interpolation matrix. Let N and n be the numbers of the elements in the column vectors on the LHS and RHS of the above equation. If the spatial resolution of an atmospheric model is coarser than an oceanic surge model, then $n<N$, and the sizes of the interpolation matrix **L** are $N\times n$.

Substituting Eqs. (43) and (44) into Eq. (21) results in

$$\boldsymbol{\eta} = \mathbf{G_{cL}} * \tilde{\mathbf{f}} \tag{45}$$

$$\tilde{\mathbf{f}} = \left[\tilde{\eta}_a \quad \tilde{\tau}_x \quad \tilde{\tau}_y\right]^T. \tag{46}$$

where

$$\underset{L_G\times n}{\mathbf{G_{cL}}} \equiv \underset{L_G\times N}{\mathbf{G_c}} \quad \underset{N\times N}{\mathbf{B}} \quad \underset{N\times n}{\mathbf{L}}. \tag{47}$$

Now, $\mathbf{G^{cL}}$ is a new convolution matrix, whose sizes are $L^G\times n$, where L^G is given in Eq. (29). The number of columns of $\mathbf{G^{cL}}$ is n, whereas the number of columns of $\mathbf{G^c}$ is N. For the surge model and MERRA model used in this study, N=32,224,425, and n=408,622. $\mathbf{G^{cL}}$ has 408,622 columns, a reduction by 31,968,881 from that of $\mathbf{G^c}$. This is a huge reduction, which helps greatly store the matrix and enhance the computational efficiency. Therefore, instead of Eq. (21), Eq. (45) should be used for storm surge simulations; its subscripts and the tilde sign may be dropped in the context where there is no ambiguity.

The left multiplication of **BL** with $\mathbf{G}_c$ does not have to wait until $\mathbf{G}_c$ is assembled. The multiplication by BL should be performed when the row vectors are about to be written to disk, as shown by the MATLAB function ASGF_conv in Appendix 4. This way, when the row vectors are loaded back to RAM, the $\mathbf{G}_{c\mathbf{L}}$ matrix can be directly assembled.

Appendix 4: MATLAB functions to calculate the ASGF matrices of $\mathbf{G}_i$ and $\mathbf{G}_c$

This appendix provides two MATLAB functions to calculate the $\mathbf{G}_i$ and $\mathbf{G}_c$ matrices that are defined by Eqs. (16) and (17), respectively. The function ASGF_ini (Table 1) calculates $\mathbf{G}_i$, whereas the function ASGF_conv (Table 2) calculates $\mathbf{G}_c$. The two functions assume that the dynamics matrix **A** has been made explicitly available. If the dynamics matrix **A** is only given in terms of its factor matrices the codes must be modified according to the algorithm given in Appendix 2.

The coding for ASGF_ini follows the algorithm given by Eqs. (22), (23), and (24) and the concept of the sampling date, d, that is discussed in Section 3.3. The code for ASGF_conv is mostly the same as that for ASGF_ini with one important difference. In ASGF_conv, the matrix **G** does not record the row vectors themselves but rather the sums of their subsets. The sums are needed because the atmospheric forcing vectors are usually given with a much larger time step compared with the time step that is required by the surge model. As a simple example, let $k_{\max}$=5 in Eq. (13). Dropping the initial condition term, we have

$$\eta^{(k+1)} = \sum_{i=0}^{k} \mathbf{r}_i \mathbf{f}^{(k-i)}, \quad k = 0,1,2,3,4,5 \tag{48}$$

Table 1 MATLAB function ASGF_ini used to calculate the ASGF matrix $\mathbf{G}_i$ for the initial value problem

```
 1  function G = ASGF_ini(A, n, kmem, d)
 2  % To calculate the ASGF matrix for the initial value problem.
 3  %
 4  % Inputs: A,  the dynamic matrix; it should be a square matrix.
 5  %         n,  the nth grid point (POI, point of interest).
 6  %      kmem, the maximum of iterations.
 7  %         d,  an integer decimating factor (>=1) to record the
 8  %             row-vectors of the powers of A into G.
 9  %
10  % Output: G, the ASGF matrix for the POI.
11  %
12
13  N=size(A,2); r=zeros(1,N); r(n)=1; M=ceil(kmem/d);
14  G=zeros(M,N);    % pre-allocate memory for G
15
16  m=0;
17  for k=1:kmem+1
18      r=r*A;
19      if mod(k,d)==0
20         m=m+1; G(m,:)=r;
21      end
22  end
23  G=G(1:m,:);
24  end  % end of the function
```

Expanding this so that we can see it more intuitively,

$$\begin{bmatrix} \eta^{(1)} \\ \eta^{(2)} \\ \eta^{(3)} \\ \eta^{(4)} \\ \eta^{(5)} \\ \eta^{(6)} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_0 & & & & & \\ \mathbf{r}_1 & \mathbf{r}_0 & & & & \\ \mathbf{r}_2 & \mathbf{r}_1 & \mathbf{r}_0 & & & \\ \mathbf{r}_3 & \mathbf{r}_2 & \mathbf{r}_1 & \mathbf{r}_0 & & \\ \mathbf{r}_4 & \mathbf{r}_3 & \mathbf{r}_2 & \mathbf{r}_1 & \mathbf{r}_0 & \\ \mathbf{r}_5 & \mathbf{r}_4 & \mathbf{r}_3 & \mathbf{r}_2 & \mathbf{r}_1 & \mathbf{r}_0 \end{bmatrix} \begin{bmatrix} \mathbf{f}^{(0)} \\ \mathbf{f}^{(1)} \\ \mathbf{f}^{(2)} \\ \mathbf{f}^{(3)} \\ \mathbf{f}^{(4)} \\ \mathbf{f}^{(5)} \end{bmatrix} \tag{49}$$

where the omitted elements in the upper triangle of the matrix are all zero. Assume that the forcing vector varies for every three time steps; i.e., $\mathbf{f}^{(0)}=\mathbf{f}^{(1)}=\mathbf{f}^{(2)}$ and $\mathbf{f}^{(3)}=\mathbf{f}^{(4)}=\mathbf{f}^{(5)}$. In addition, assume that we only wish to retain the solutions at every three steps. Equation (49) can then be reduced to

$$\begin{bmatrix} \eta^{(3)} \\ \eta^{(6)} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_0+\mathbf{r}_1+\mathbf{r}_2 & \mathbf{0} \\ \mathbf{r}_3+\mathbf{r}_4+\mathbf{r}_5 & \mathbf{r}_0+\mathbf{r}_1+\mathbf{r}_2 \end{bmatrix} \begin{bmatrix} \mathbf{f}^{(0)} \\ \mathbf{f}^{(3)} \end{bmatrix}. \tag{50}$$

Because the forcing vector varies with every three time steps, we have reduced a 6×6 matrix to a 2×2 matrix. The elements in the lower triangle of the reduced matrix are the sums of every three row vectors (i.e., $\mathbf{r}_0+\mathbf{r}_1+\mathbf{r}_2$ and $\mathbf{r}_3+\mathbf{r}_4+\mathbf{r}_5$). In this simple example, the decimating factor d is 3. For a realistic case, let us assume that the value of Δt that is required by the surge model is 5 sec, whereas the atmospheric forcing vectors are given hourly. The gap from 5 to 3600 sec represents a factor of 720 to sum over the row vectors and to reduce the size of the matrix **G**. In the function ASGF_conv, the vector **g** holds the accumulation of the **r**-vectors. The vector **g** is periodically (every d steps) recorded into the matrix **G** and then renewed to the current value of the **r** vector before a new round of accumulation starts. The recording and renewal are performed in line 29, and the accumulation is performed in line 31. The last two optional inputs to the function are **B** and **L**, which were discussed in Appendix 3. If they are input, the vector **g** will be multiplied by **BL** before it is recorded into **G**. Thus, the columns of **G** can be reduced if **L** is a tall and thin matrix.

In both ASGF_ini and ASGF_conv, the G matrix is assembled during the calculations of the **r**-vectors. That they are so coded is more for the purpose of showing the link between the **r**-vectors and the **G** matrix ($\mathbf{G}_i$ or $\mathbf{G}_c$) and how the former can be assembled into the latter. In practice, especially when the size of the problem is large, we should separate the calculation of the **r**-vectors and their assembly into two processes for the sake of economizing the RAM usage. The calculation of the **r**-vectors can proceed along with periodically writing the vectors into disk files. The assembly can be later performed (on demand) when loading the vectors from the disk files.

Table 2 MATLAB function ASGF_conv used to calculate the ASGF matrix $\mathbf{G}_c$ for the forced problem

```
function G = ASGF_conv(A, n, kmem, d, B, L)
% To calculate the ASGF matrix for the forced problem.
%
% Inputs: A,    the dynamic matrix; it should be a square
%               matrix.
%         n,    the nth grid point (POI, point of interest).
%         kmem, the maximum number of iterations.
%         d,    an integer decimating factor (>=1) to record the
%               row-vectors of the powers of A into G.
%         B,    optional input, for mapping the external force
%               into the momentum to change the state vector of
%               the system.
%         L,    optional input, which interpret the forcing
%               field as given to the grid used by the dynamic
%               system.
% Output: G, the ASGF matrix for the POI.
%
N=size(A,2); r=zeros(1,N); r(n)=1; g=r; M=ceil(kmem/d);

if nargin==5, BL=B;   N=size(BL,2); end
if nargin==6, BL=B*L; N=size(BL,2); end
G=zeros(M,N);   % pre-allocate memory for G

m=0;
for k=1:kmem+1
    r=r*A;
    if mod(k,d)==0
       if nargin>=5, g=g*BL; end
       m=m+1; G(m,:)=g; g=r;
    else
       g=g+r;
    end
end
G=G(1:m,:);
end  % end of the function
```

Appendix 5: computational loads of the ADI scheme

This appendix analyses the computational load of the ADI method. The result of the analysis from this appendix will be used in part II of this study, where the computational efficiencies of the ASGF method and the traditional method will be compared, and the ADI method will be used to represent the traditional method of modeling storm surges. Analyzing the computational load of the ADI scheme is a complicated task because it involves implicit solutions. To render the task simpler, Cartesian coordinates are adopted for the analysis.

We need to recall the well-known C-grid (Arakawa and Lamb 1977), which was used in this study. Figure 7 shows a C-grid with I-horizontal lines (parallel to the x-axis) and J-vertical lines (parallel to the y-axis). The η-points are arranged at the intersections of the even-numbered lines (i.e., $i=2,4,\cdots; I-1$, and $j=2,4,\cdots;J-1$). I and J are chosen to be odd numbers so that $i=1$, $i=I$, $j=1$, and $j=J$ can be the coastal lines bounding the model domain. There are $(J-1)/2$ number of η variables on an even-numbered horizontal line, and there are $(I\text{-}1)/2$ such horizontal lines. The total number of η variables, N_e, is therefore

$$N_e = \frac{I-1}{2}\frac{J-1}{2}. \tag{51}$$

In addition, there are $(J-1)/2-1$ U variables on an even-numbered horizontal line, excluding $U_{i,1}$ and $U_{i,J}$ (they are constants, always equal to zero at the two coastal ends). The total number of U variables over the model domain is then

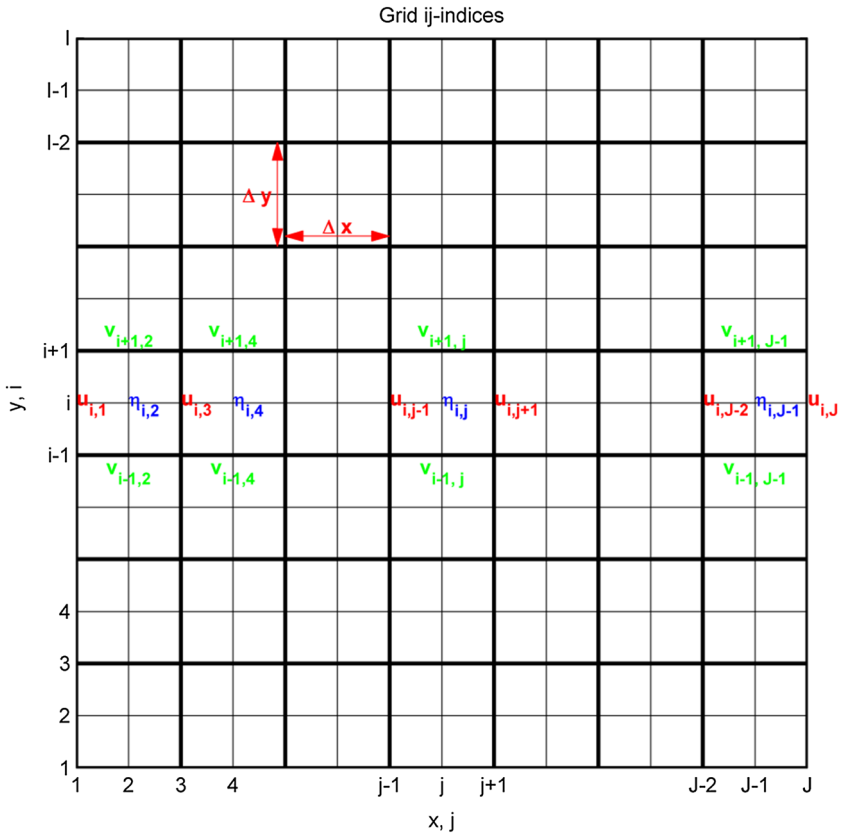

Fig. 7 C-grid with *I*-horizontal lines and *J*-vertical lines, where *I* and *J* are odd numbers with $i=1$, $i=I$, $j=1$, and $j=J$ for the coastal lines bounding the model domain. The *odd-numbered lines* form the sides of the cells. The intersections of the *even-numbered lines* are the centers of the cells. The model variables, η, U, and V are annotated on the cells along ith-horizontal line to illustrate how they are arranged on the grid points

$$N_u = \frac{I-1}{2}\left(\frac{J-1}{2}-1\right). \tag{52}$$

In the same way, we can find that the total number of v variables is

$$N_v = \frac{J-1}{2}\left(\frac{I-1}{2}-1\right). \tag{53}$$

We can see that when I and J are both large, $N_e/N_u \approx 1 \approx N_e/N_v$. The total number of model variables is the sum of the three parts,

$$N = N_e + N_u + N_v. \tag{54}$$

We can now proceed to examine the computational load of the ADI scheme. As noted in Appendix 1 of part I, the ADI scheme splits each time step into two halves. For the first half-time step, η and U variables along the horizontal grid lines are treated implicitly. For the second half-time step, η and V variables along the vertical grid lines are treated implicitly. Referring to Fig. 7, we can write out the tri-diagonal matrix equation for the η and U variables along the ith line ($i=2,4,\cdots,I-1$) as

$$\begin{bmatrix} 1 & s_x & & & & \\ -s_x g h_{i,3} & \beta_{i,3} & s_x g h_{i,3} & & & \\ & -s_x & 1 & s_x & & \\ & & \ddots & \ddots & \ddots & \\ & & & -s_x g h_{i,J-1} & \beta_{i,J-1} & s_x g h_{i,J-1} \\ & & & & -s_x & 1 \end{bmatrix} \begin{bmatrix} \eta_{i,2} \\ U_{i,3} \\ \eta_{i,4} \\ \vdots \\ U_{i,J-2} \\ \eta_{i,J-1} \end{bmatrix}^{(k+1/2)} = \begin{bmatrix} \eta_{i,2} \\ U_{i,3} \\ \eta_{i,4} \\ \vdots \\ U_{i,J-2} \\ \eta_{i,J-1} \end{bmatrix}^{(k)} + \mathbf{R}_2^{(k)} + \mathbf{R}_3^{(k)} \tag{55}$$

where

$$\mathbf{R}_2^{(k)} = \begin{bmatrix} s_y\left(V_{i-1,2}-V_{i+1,2}\right)^{(k)} \\ f_{i,3}\bar{V}_{i,3}^{(k)} \\ s_y\left(V_{i-1,4}-V_{i+1,4}\right)^{(k)} \\ \vdots \\ f_{i,J-2}\bar{V}_{i,J-2}^{(k)} \\ s_y\left(V_{i-1,J-1}-V_{i+1,J-1}\right)^{(k)} \end{bmatrix} \tag{56}$$

$$\mathbf{R}_3^{(k)} = \begin{bmatrix} 0 \\ s_x g h_{i,3}\left(\eta^a_{i,2}-\eta^a_{i,4}\right)^{(k)} + 0.5\Delta t\tau^x_{i,3}\Big|^{(k)} \\ 0 \\ \vdots \\ s_x g h_{i,J-2}\left(\eta^a_{i,J-2}-\eta^a_{i,J-2}\right)^{(k)} + 0.5\Delta t\tau^x_{i,J-2}\Big|^{(k)} \\ 0 \end{bmatrix} \tag{57}$$

and $\beta_{i,j}=1+0.5\Delta t k_{i,j}/h_{i,j}$, $s_x=0.5\Delta t/\Delta x$, $s_y=0.5\Delta t/\Delta y$, $\mathbf{R}_2{}^{(k)}$ contains the mass divergence in the y-direction and the Coriolis forcing in the x-direction, and $\mathbf{R}_3{}^{(k)}$ contains the atmospheric forcing in the x-direction. The V with bar indicates the averaged value of four V's surrounding the (i,j)-point.

We can now count how many multiplications are needed to solve Eq. (55). The two column vectors on the RHS of the equation have to be evaluated first. Eq. (56) shows that to evaluate $\mathbf{R}_2{}^{(k)}$ requires $(\frac{J-1}{2}+\frac{J-1}{2}-1)$ multiplications, and Eq. (57) shows that to evaluate $\mathbf{R}_2{}^{(k)}$ requires $2\left(\frac{J-1}{2}-1\right)$ multiplications. Adding the resultant $\mathbf{R}_2{}^{(k)}$ and $\mathbf{R}_3{}^{(k)}$ to the first column vector on the RHS of the equation, we have the tri-diagonal matrix equation to solve. Based on the Thomas (1949) tri-diagonal matrix algorithm (also see, e.g., CFD Online (http://www.cfd-online.com/Wiki/Tridiagonal_matrix_algorithm_-_TDMA_(Thomas_algorithm)), solving a tridiagonal system with M-unknowns requires $5M-4$ multiplications. In our case, $M=\frac{J-1}{2}+\frac{J-1}{2}-1$. There are $(I-1)/2$ even-numbered horizontal lines, and Eq. (55) is written for only one of them. Therefore, the total number of multiplications required to solve $(I-1)/2$ equations such as Eq. (55) is

$$\begin{aligned} &= \underbrace{\left\{\underbrace{\left(\frac{J-1}{2}+\frac{J-1}{2}-1\right)}_{\text{for } R_2}+\underbrace{2\left(\frac{J-1}{2}-1\right)}_{\text{for } R}+\underbrace{5\left(\frac{J-1}{2}+\frac{J-1}{2}-1\right)-4}_{\text{for the tri-diagonal system}}\right\}}_{\text{per even-numbered horizontal line}} \times \underbrace{\frac{I-1}{2}}_{\substack{\text{number of}\\ \text{the lines}}} \\ &= (N_e+N_u)+2N_u+5(N_e+N_u)-4 \\ &= 6N_e+8N_u-4\,. \end{aligned} \tag{58}$$

Having obtained $\boldsymbol{\eta}^{(k+1/2)}$ and $\mathbf{U}^{(k+1/2)}$, we can update $\mathbf{V}^{(k+1/2)}$ according to the following equation:

$$\begin{aligned} V_{i,j}^{(k+1/2)} = &\left(1-0.5\Delta t\ \kappa/h_{i,j}\right)V_{i,j}^{(k)} - gh_{ij}s_y \\ &\times\left[\left(\eta_{i+1,j}-\eta_{i-1,j}\right)^{(k+1/2)}-\left(\eta^a_{i+1,j}-\eta^a_{i-1,j}\right)^{(k)}\right] \\ &-0.5\Delta t\ f_{i,j}\bar{U}_{i,j}^{\left(k+\frac{1}{2}\right)}+0.5\Delta t\tau^y_{i,j}\Big|^{(k)} \end{aligned} \tag{59}$$

for $i=3,5,7,\cdots,I-2$, and $j=2,4,6,\cdots,J-1$. The U with bar indicates the averaged value of four U's surrounding the (i,j)-point. This equation indicates that updating an individual element of $\mathbf{V}^{(k+1/2)}$ requires four multiplications. Because there are N_v such elements, the total number of multiplications required to update $\mathbf{V}^{(k+1/2)}$ is $4N_v$

Thus, for the first half-time step, the number of multiplications, "#*"$_{1/2}$, is

$$''\#^*''_{1/2} = 6N_e+8N_u+4N_v-4. \tag{60}$$

Similarly, the number of multiplications required in the second half-time step, "#*"$_{2/2}$, can be found as

$$''\#^*''_{2/2} = 6N_e+8N_v+4N_u-4. \tag{61}$$

The number of multiplications in one time step, "#*", is

$$\begin{aligned} ''\#^*'' = ''\#^*''_{1/2}+''\#^*''_{2/2} &= 12N_e+12N_u+12N_v-8 = \left(12-\frac{8}{N}\right)N, \\ &\approx 12N,\ \ when\ N\ is\ \text{large}. \end{aligned} \tag{62}$$

The grid shown in Fig. 7 does not consider the existence of islands or the irregularity of the coastal lines. When these complexities have to be included, the simple relations between the number of the variables and number of the grid lines, i.e., Eqs. (51), (52) and (53), become invalid. However, the number of multiplications per time step as expressed by Eqs. (60)−(61) remain valid because they are expressed in terms of the number of unknowns. We can still use them as long as we can supply the number of unknowns, and we can easily create a program to count the unknowns in this case.

References

Arakawa A, Lamb VR (1977) Computational design of the basic dynamical process of the UCLA general circulation model. Methods in Computational Phsics 17:173–265, Academic Press

Courant R, Friedrichs K, Lewy H (1967) On the partial difference equations of mathematical physics. IBM J Res Dev 11(2):215–234, MR 0213764,Zbl 0145.40402

Csanady GT (1982) Circulation in the coastal ocean. Reidel, Dordrecht, 279pp

Ding Y, Jia Y, ASCE M, Wang SSY, ASCE F (2004) Identification of Manning's roughness coefficients in shallow water flows. J Hydraul Eng ASCE

Durran DR (1999) Numerical methods for wave equations in geophysical fluid dynamics, vol 32. Springer, New York

Heaps NS (1969) A two-dimensional numerical sea model. Philos Trans R Soc London, Ser A 265(1160):93–137

Lamb H (1932) Hydrodynamics. Cambridge university press

Laplace PS (1776) Suite des recherches sur plusieurs points du systeme du monde (XXV–XXVII). Mém. Présentés Acad. R. Sci. Inst. France, 542–552

Leendertse JJ (1967) Aspects of a computational model for long-period water-wave propagation. Rand Corporation, Santa Monica, p 165

Munk WH, Cartwright DE (1966) Tidal spectroscopy and prediction. Philos Trans R Soc London, Ser A 259(1105):533–581

Pedlosky J (1979) Geophysical fluid dynamics. Springer Verlag, Berlin

Proudman J (1954) Note on the dynamical theory of storm-surges. Arch Meteorol Geophys Bioklimatol A 7(1):344–351

Randall (2007) The Laplace tidal equations and atmospheric tides. Available from http://kiwi.atmos.colostate.edu/group/dave/pdf/LTE.frame.pdf.

Schwab DJ (1978) Simulation and forecasting of Lake Erie storm surges. Mon Weather Rev 106(10):1476–1487

Shuto N (1991) Numerical simulation of tsunamis—its present and near future. Nat Hazards 4:171–191

Sielecki A (1968) An energy-conserving difference scheme for the storm surge equations. Mon Weather Rev 96:150–156

Svansson A (1959) Some computations of water heights and currents in the Baltic. Tellus 11(2):231–238

Tan WY (1992) Shallow water hydrodynamics: mathematical theory and numerical solution for a two-dimensional system of shallow-water equations. Elsevier, Amsterdam

Thomas, L.H. (1949). Elliptic problems in linear differential equations over a network. Watson Sci. Comput. Lab Report, Columbia University, New York

Uusitalo S (1960) The numerical calculation of wind effect on sea level elevations. Tellus 12(4):427–435

Welander P (1961) Numerical prediction of storm surges. Adv Geophys 8:315

Wesseling P (2009) Principles of computational fluid dynamics, vol 29. Springer Science & Business, Berlin

Xu Z (2007) The all-source Green's function and its applications to tsunami problems. Sci Tsunami Haz 26(1):59–69

Xu Z (2011) The all-source Green's function of linear shallow water dynamic system: its numerical constructions and applications to tsunami problems. The Tsunami Threat-Res Technol 25

Xu Z (2015) The all-source Green's function (ASGF) and its applications to storm surge modeling, part II: from the ASGF Convolution to Forcing Data Compression and A Regression Model. Ocean Dyn. doi:10.1007/s10236-015-0894-y

Xu Z, Song YT (2013) Combining the all-source Green's functions and the GPS-derived source functions for fast tsunami predictions—illustrated by the March 2011 Japan tsunami. J Atmos Ocean Technol 30(7):1542–1554

Xu Z, Savard J-P, Lefaivre D (2015) Data assimilative hindcast and climatological forecast of storm surges at Sept-Îles in the estuary of gulf of St. Lawrence. Atmosphere–Ocean. doi:10.1080/07055900.2015.1079774